



National Textile University

Department of Computer Science

Subject:

Operating System

Submitted to:

Sir Nasir

Submitted by:

Eman Babar

Reg. number:

23-NTU-CS-FL-1148

Semester:

5th- A

LAB-10-HomeTask

Task1: Hotel Rom Occupancy Problem

Code:

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <pthread.h>
 4 #include <unistd.h>
 5 #include <semaphore.h>
 6 #define Total_People 10
 7 sem_t room_semaphore;
 8 pthread_mutex_t print_lock;
 9 int occupied_rooms = 0;
10 void* hotel_guest(void* arg){
11     int personID = *(int*)arg;
12     sem_wait(&room_semaphore);
13     pthread_mutex_lock(&print_lock);
14     occupied_rooms++;
15     printf("Person %d entered. Rooms filled: %d\n", personID, occupied_rooms);
16     pthread_mutex_unlock(&print_lock);
17     sleep(1+rand()%3);
18     pthread_mutex_lock(&print_lock);
19     occupied_rooms--;
20     printf("Person %d leaving. Rooms filled: %d\n", personID, occupied_rooms);
21     pthread_mutex_unlock(&print_lock);
22     sem_post(&room_semaphore);
23     return NULL;
24 }
25 int main(){
26     int N;
27     printf("Enter number of rooms in the hotel: ");
28     scanf("%d", &N);
29     sem_init(&room_semaphore, 0, N);
30     pthread_mutex_init(&print_lock, NULL);
31     pthread_t guests[Total_People];
32     int personIDs[Total_People];
33     srand(time(NULL));
34     for(int i=0; i<Total_People; i++){
35         personIDs[i] = i+1;
36         pthread_create(&guests[i], NULL, hotel_guest, &personIDs[i]);
37     }
38     for(int i=0; i<Total_People; i++){
39         pthread_join(guests[i], NULL);
40     }
41     sem_destroy(&room_semaphore);
42     pthread_mutex_destroy(&print_lock);
43     return 0;
44 }
```

Output:

Task2: Download Manager Simulation

Code:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <unistd.h>
5 #include <semaphore.h>
6 #define Total_Downloads 8
7 sem_t download_slots;
8 pthread_mutex_t lock;
9 void* download_file(void* arg){
10     int id = *(int*)arg;
11     sem_wait(&download_slots);
12     pthread_mutex_lock(&lock);
13     printf("Download %d started.\n", id);
14     pthread_mutex_unlock(&lock);
15     sleep(1 + rand() % 5);
16     pthread_mutex_lock(&lock);
17     printf("Download %d completed.\n", id);
18     pthread_mutex_unlock(&lock);
19     sem_post(&download_slots);
20     return NULL;
21 }
22 int main(){
23     srand(time(NULL));
24     sem_init(&download_slots, 0, 3);
25     pthread_mutex_init(&lock, NULL);
26     pthread_t downloads[Total_Downloads];
27     int downloadIDs[Total_Downloads];
28     for(int i = 0; i < Total_Downloads; i++){
29         downloadIDs[i] = i + 1;
30         pthread_create(&downloads[i], NULL, download_file, &downloadIDs[i]);
31     }
32     for(int i = 0; i < Total_Downloads; i++){
33         pthread_join(downloads[i], NULL);
34     }
35     sem_destroy(&download_slots);
36     pthread_mutex_destroy(&lock);
37     return 0;
38 }
```

Output:

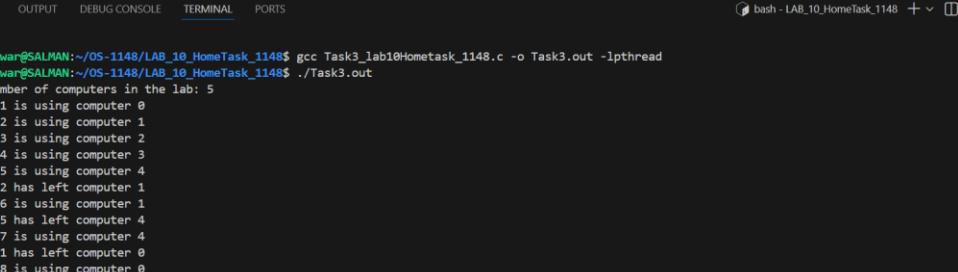
```
● ohai_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ gcc Task2_lab10Hometask_1148.c -o Task2.out -lpthread
● ohai_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ ./Task2.out
Download 1 started.
Download 2 started.
Download 3 started.
Download 3 completed.
Download 4 started.
Download 4 completed.
Download 5 started.
Download 1 completed.
Download 2 completed.
Download 6 started.
Download 7 started.
Download 5 completed.
Download 6 completed.
Download 8 started.
Download 8 completed.
Download 7 completed.
○ ohai_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$
```

Task3: Library Computer Lab Access

Code:

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <pthread.h>
 4 #include <unistd.h>
 5 #include <semaphore.h>
 6 #define Total_Students 10
 7 sem_t available_computers;
 8 pthread_mutex_t array_lock;
 9 int *computer_users;
10 int K;
11 void* student(void* arg){
12     int id = *(int*)arg;
13     sem_wait(&available_computers);
14     int assigned_computer = -1;
15     pthread_mutex_lock(&array_lock);
16     for(int i = 0; i < K; i++){
17         if(computer_users[i] == 0){
18             computer_users[i] = id;
19             assigned_computer = i;
20             printf("Student %d is using computer %d\n", id, i);
21             break;
22         }
23     }
24     pthread_mutex_unlock(&array_lock);
25     sleep(1 + rand() % 4);
26     pthread_mutex_lock(&array_lock);
27     computer_users[assigned_computer] = 0;
28     printf("Student %d has left computer %d\n", id, assigned_computer);
29     pthread_mutex_unlock(&array_lock);
30     sem_post(&available_computers);
31     return NULL;
32 }
33 int main(){
34     srand(time(NULL));
35     printf("Enter number of computers in the lab: ");
36     scanf("%d", &K);
37     computer_users = (int*)calloc(K, sizeof(int));
38     sem_init(&available_computers, 0, K);
39     pthread_mutex_init(&array_lock, NULL);
40     pthread_t students[Total_Students];
41     int studentIDs[Total_Students];
42     for(int i = 0; i < Total_Students; i++){
43         studentIDs[i] = i + 1;
44         pthread_create(&students[i], NULL, student, &studentIDs[i]);
45     }
46     for(int i = 0; i < Total_Students; i++){
47         pthread_join(students[i], NULL);
48     }
49     free(computer_users);
50     sem_destroy(&available_computers);
51     pthread_mutex_destroy(&array_lock);
52     return 0;
53 }
```

Output:



```
C Task1_lab10Hometask_1148.c U C Task2_lab10Hometask_1148.c U C Task3_lab10Hometask_1148.c U
C Task3_lab10Hometask_1148.c > (main)
34 |     srand(time(NULL));
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash - LAB_10_HomeTask_1148 + - | x

● ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ gcc Task3_lab10Hometask_1148.c -o Task3.out -lpthread
● ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ ./Task3.out
Enter number of computers in the lab: 5
Student 1 is using computer 0
Student 2 is using computer 1
Student 3 is using computer 2
Student 4 is using computer 3
Student 5 is using computer 4
Student 2 has left computer 1
Student 6 is using computer 1
Student 5 has left computer 4
Student 7 is using computer 4
Student 1 has left computer 0
Student 8 is using computer 0
Student 4 has left computer 3
Student 9 is using computer 3
Student 7 has left computer 4
Student 10 is using computer 4
Student 6 has left computer 1
Student 3 has left computer 2
Student 10 has left computer 4
Student 8 has left computer 0
Student 9 has left computer 3
● ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$
```

Task4: Thread Pool/ Worker Pool Simulation

Code:

```
● ● ●  
1 #include <stdio.h>  
2 #include <stdlib.h>  
3 #include <pthread.h>  
4 #include <unistd.h>  
5 #include <semaphore.h>  
6 #define Total_tasks 10  
7 #define Workers 3  
8 sem_t worker_slots;  
9 pthread_mutex_t print_lock;  
10 void* run_task(void* arg){  
11     int id = *(int*)arg;  
12     sem_wait(&worker_slots);  
13     pthread_mutex_lock(&print_lock);  
14     printf("Task %d is being started(worker assigned).\n", id);  
15     pthread_mutex_unlock(&print_lock);  
16     sleep(1 + rand() % 2);  
17     pthread_mutex_lock(&print_lock);  
18     printf("Task %d has been completed(worker released).\n", id);  
19     pthread_mutex_unlock(&print_lock);  
20     sem_post(&worker_slots);  
21     return NULL;  
22 }  
23 int main(){  
24     srand(time(NULL));  
25     sem_init(&worker_slots, 0, Workers);  
26     pthread_mutex_init(&print_lock, NULL);  
27     pthread_t threads[Total_tasks];  
28     int taskIDs[Total_tasks];  
29     for(int i = 0; i < Total_tasks; i++){  
30         taskIDs[i] = i + 1;  
31         pthread_create(&threads[i], NULL, run_task, &taskIDs[i]);  
32     }  
33     for(int i = 0; i < Total_tasks; i++){  
34         pthread_join(threads[i], NULL);  
35     }  
36     sem_destroy(&worker_slots);  
37     pthread_mutex_destroy(&print_lock);  
38     return 0;  
39 }  
40
```

Output:

```
Task4_lab10Hometask_1148.c > ...
10 void* run_task(void* arg){
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ gcc Task4_lab10Hometask_1148.c -o Task4.out -lpthread
● ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ ./Task4.out
Task 1 is being started(worker assigned).
Task 3 is being started(worker assigned).
Task 2 is being started(worker assigned).
Task 1 has been completed(worker released).
Task 4 is being started(worker assigned).
Task 2 has been completed(worker released).
Task 5 is being started(worker assigned).
Task 3 has been completed(worker released).
Task 6 is being started(worker assigned).
Task 4 has been completed(worker released).
Task 7 is being started(worker assigned).
Task 5 has been completed(worker released).
Task 8 is being started(worker assigned).
Task 6 has been completed(worker released).
Task 9 is being started(worker assigned).
Task 7 has been completed(worker released).
Task 10 is being started(worker assigned).
Task 8 has been completed(worker released).
Task 9 has been completed(worker released).
Task 10 has been completed(worker released).
○ ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$
```

WSL: Ubuntu-24.04 | main* | 171 101 | 0 0 0 | 04/12/2023 1:18 pm

Task5: Car Wash Station

Code:

```
● ● ●
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <unistd.h>
5 #include <semaphore.h>
6 #define Total_Cars 8
7 sem_t wash_stations;
8 pthread_mutex_t print_lock;
9 void* car_wash(void* arg){
10     int carID = *(int*)arg;
11     sem_wait(&wash_stations);
12     pthread_mutex_lock(&print_lock);
13     printf("Car %d entered the wash station.\n", carID);
14     pthread_mutex_unlock(&print_lock);
15     sleep(3);
16     pthread_mutex_lock(&print_lock);
17     printf("Car %d has left the wash station.\n", carID);
18     pthread_mutex_unlock(&print_lock);
19     sem_post(&wash_stations);
20     return NULL;
21 }
22 int main(){
23     srand(time(NULL));
24     sem_init(&wash_stations, 0, 2);
25     pthread_mutex_init(&print_lock, NULL);
26     pthread_t cars[Total_Cars];
27     int carIDs[Total_Cars];
28     for(int i = 0; i < Total_Cars; i++){
29         carIDs[i] = i + 1;
30         pthread_create(&cars[i], NULL, car_wash, &carIDs[i]);
31     }
32     for(int i = 0; i < Total_Cars; i++){
33         pthread_join(cars[i], NULL);
34     }
35     sem_destroy(&wash_stations);
36     pthread_mutex_destroy(&print_lock);
37     return 0;
38 }
```

Output:

The screenshot shows a Windows 10 desktop environment with the Visual Studio Code application open. The title bar indicates the workspace is named 'LAB_10_Hometask_1148' and it's running in WSL: Ubuntu-24.04. The left sidebar contains icons for file operations like New File, Open, Save, Find, Replace, and others. The main area has five tabs for C files: 'ask_1148.c', 'Task2_lab10Hometask_1148.c', 'Task3_lab10Hometask_1148.c', 'Task4_lab10Hometask_1148.c', and 'Task5_lab10Hometask_1148.c'. Below the tabs are buttons for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected, showing a terminal window with the following content:

```
● ohai1_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ gcc Task5_lab10Hometask_1148.c -o Task5.out -lpthread
● ohai1_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ ./Task5.out
Car 1 entered the wash station.
Car 2 entered the wash station.
Car 1 has left the wash station.
Car 2 has left the wash station.
Car 4 entered the wash station.
Car 3 entered the wash station.
Car 4 has left the wash station.
Car 3 has left the wash station.
Car 5 entered the wash station.
Car 6 entered the wash station.
Car 6 has left the wash station.
Car 7 entered the wash station.
Car 5 has left the wash station.
Car 8 entered the wash station.
Car 7 has left the wash station.
Car 8 has left the wash station.
● ohai1_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$
```

Task6: Traffic Bridge Control(Single-Lane Bridge)

Code:



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <unistd.h>
5 #include <semaphore.h>
6 #define Total_Cars 10
7 sem_t bridge_slots;
8 pthread_mutex_t print_lock;
9 void* car(void* arg){
10     int id = *(int*)arg;
11     sem_wait(&bridge_slots);
12     pthread_mutex_lock(&print_lock);
13     printf("Car %d is crossing the bridge.\n", id);
14     pthread_mutex_unlock(&print_lock);
15     sleep(1 + rand() % 4);
16     pthread_mutex_lock(&print_lock);
17     printf("Car %d has crossed the bridge.\n", id);
18     pthread_mutex_unlock(&print_lock);
19     sem_post(&bridge_slots);
20     return NULL;
21 }
22 int main(){
23     srand(time(NULL));
24     sem_init(&bridge_slots, 0, 3);
25     pthread_mutex_init(&print_lock, NULL);
26     pthread_t cars[Total_Cars];
27     int carIDs[Total_Cars];
28     for(int i = 0; i < Total_Cars; i++){
29         carIDs[i] = i + 1;
30         pthread_create(&cars[i], NULL, car, &carIDs[i]);
31     }
32     for(int i = 0; i < Total_Cars; i++){
33         pthread_join(cars[i], NULL);
34     }
35     sem_destroy(&bridge_slots);
36     pthread_mutex_destroy(&print_lock);
37     return 0;
38 }
```

Output:

The screenshot shows a terminal window in VS Code displaying the output of a C program. The program solves a variation of the classic bridge crossing puzzle where 100 llamas need to cross a river using a boat that can only carry 2 llamas at a time. The output shows the sequence of crossings:

```
Car 1 is crossing the bridge.  
Car 2 is crossing the bridge.  
Car 3 is crossing the bridge.  
Car 2 has crossed the bridge.  
Car 3 has crossed the bridge.  
Car 4 is crossing the bridge.  
Car 5 is crossing the bridge.  
Car 1 has crossed the bridge.  
Car 6 is crossing the bridge.  
Car 5 has crossed the bridge.  
Car 7 is crossing the bridge.  
Car 6 has crossed the bridge.  
Car 8 is crossing the bridge.  
Car 4 has crossed the bridge.  
Car 9 is crossing the bridge.  
Car 8 has crossed the bridge.  
Car 10 is crossing the bridge.  
Car 7 has crossed the bridge.  
Car 9 has crossed the bridge.  
Car 10 has crossed the bridge.
```